Programmation Python

De la base aux concepts avancés de Python

Audrey Kouessi FANGNON & Johannès HOUNSINOU

Bénin Excellence - Fondation Vallet / FSS COTONOU

9 novembre 2024

Plan du cours

1. Chapitre 3 : Les instructions de contrôle de flux Python.
   - Les structures conditionnelles « if », « else », « elif »
   - Les instructions répétitives « while » et « for »
   - Les instructions « pass », « break », « continue »

Les instructions de contrôle de flux Python

Définition

Les instructions de contrôle permettent de controler le flux d'execution du programme en déterminant l'ordre dans lequel les instructions sont exécutées. Dans le langage Python, il existe principalement deux types de structure de controle à savoir :

- Les structures conditionnelles (if, else, elif)
- Les boucles (for, while)

L'instruction conditionnelle « if »

Définition

L'instruction if (si en français) permet d'exécuter certaines instructions uniquement si une condition est vérifiée. Comme le traitement va dépendre d'une condition, cette structure est qualifiée d'instruction conditionnelle if.

Usage

```
if condition :
    bloc d'instructions
```

L'instruction conditionnelle « if »

Exemple :

```
1  moyenne_bac_ecrit = float(input("Entrez votre
   moyenne à l'écrit au BAC : "))
2  if moyenne_bac_ecrit >= 9 :
3      print("Bravo. Vous êtes admissible au BAC.")
```

Indentation

L'indentation est le retrait de certaines lignes de code afin de délimiter les blocs d'exécution. Il est recommandé d'utiliser soit la touche tabulation, soit quatre (04) espaces pour une indentation.

Python n'accepte pas le mélange des deux formes d'indentation dans un même code.

La structure conditionnelle « if ... else »

if ... else

Un else (sinon en français) peut etre ajouté à la suite de l'instruction if pour rendre le code plus complet.

Usage

```
if condition :
    bloc d'instructions 1
else:
    bloc d'instructions 2
```

La structure conditionnelle « if ... else »

Exemple :

```
1  moyenne_bac_ecrit = float(input("Entrez votre
   moyenne à l'écrit au BAC : "))
2  if moyenne_bac_ecrit >= 9 :
3      print("Bravo. Vous êtes admissible au BAC.")
4  else :
5      print("Navré! Revenez l'année prochaine.")
```

La structure conditionnelle « if ... elif ... else »

if ... elif ... else

Le mot clé elif est une contraction de "else if" (sinon si) et est utilisé pour introduire une condition supplémentaire dans une structure conditionnelle.

```
if condition_1 :
    bloc d'instructions 1
elif condition_2 :
    bloc d'instructions 2
elif condition_3 :
    bloc d'instructions 3
...
else:
    bloc d'instructions final
```

La structure conditionnelle « if ... elif ... else »

Exemple :

```
1  moyenne_bac_ecrit = float(input("Entrez votre moyenne à l'écrit au BAC : "))
2  if moyenne_bac_ecrit >= 9 :
3      print("Bravo! Vous êtes admissible au BAC.")
4  elif moyenne_bac_ecrit < 5 :
5      print("Navré. Vous serez exlut au BAC pour insuffisance de moyenne")
6  else :
7      print("Navré! Revenez l'année prochaine.")
```

Pour aller loin

if imbriqué

Un code Python peut comporter des if imbriqués par exemple pour distinguer des sous cas. Exemple.

```
if a > 0 :
    print("cas 1")
    if b > 0
        print("sous-cas 1")
    else :
        print("sous-cas 2")
else :
    print("cas 2")
```

Pour aller loin

L'opérateur ternaire

L'opérateur ternaire est une manière concise d'exprimer une structure conditionnelle en une seule ligne.

`<expression_1> if <condition> else <expression_2>`

Les deux codes ci-dessous sont equivalents.

```
1 age = int(input("Entrez votre age: "))
2 print("Majeur") if age >= 18 else print("Mineur")
```

```
1 age = int(input("Entrez votre age: "))
2 if age >= 18 :
3     print("Majeur")
4 else:
5     print("Mineur")
```

Coding Time - La structure conditionnelle « if ... elif ... else »

Problème A - signum

Ecrivez un programme qui demande à l'utilisateur un nombre et indique s'il est positif, négatif ou nul.

Problème B - IMC Indice de Masse Corporelle

Ecrivez un programme qui demande à l'utilisateur de saisir son poids (en Kg) et sa taille (en m) puis calcule l'IMC (*selon l'OMS,* $IMC = \frac{Poids}{Taille^2}$), affiche et interprète le resultat.

Interprétation	Sous-poids	Normal	Surpoids	Obésité
IMC (kg/m^2)	< 18.5	18.5 à 25^-	25 à 30^-	$\geqslant 30$

Table 1 – Intéprétation IMC `www.wikipedia.org`

Coding Time - La structure conditionnelle « if ... elif ... else »

Problème C - Equation $ax^2 + bx + c = 0$.

Ecrivez un programme qui résous les équations de la forme : $ax^2 + bx + c = 0$ dans $\mathbb{R}$ où $a \in \mathbb{R}^*$, $b \in \mathbb{R}$ et $c \in \mathbb{R}$.Le programme :

1. lit les nombres réels a, b et c au clavier.
2. si $a \neq 0$ calcule le discriminant $\Delta = b^2 - 4ac$ sinon avertit l'utilisateur et arrête le programme.
3. imprime l'équation $ax^2 + bx + c = 0$ correspondant.
4. Interprète le résultat et affiche les solutions.
   - $\Delta < 0$: pas de solution dans $\mathbb{R}$
   - $\Delta = 0$: solution double $x_0 = \frac{-b}{2a}$
   - $\Delta > 0$: deux solutions réelles distinctes $x_1 = \frac{-b-\sqrt{\Delta}}{2a}$ et $x_2 = \frac{-b+\sqrt{\Delta}}{2a}$

L'instruction « while »

La boucle while

La boucle while (tant que en français) est utilisée si on désir répéter plusieurs fois une même série d'instructions tant qu'une condition est vraie. La syntaxe est :

```
while condition :
    bloc d'instructions
```

Remarque

- Si la `condition` est vraie et ne devient jamais faux, la boucle ne s'arretera jamais. On dit que c'est une boucle infinie.
- Si la `condition` est fausse au début, la boucle ne s'exécutera jamais.

L'instruction « while »

Exemple 1

```
1 x = 0
2 while x < 10 :
3     print(x)
4     x += 1
```

Exemple 2

```
1 """ Je compte de x à 100 """
2 x = int(input("Entrez un entier compris entre 0 et
  100 : "))
3 while x < 100 :
4     print(x)
5     x += 1
```

Coding Time - L'instruction « while »

Problème A - Diviseur positif

Écrivez un programme qui demande à l'utilisateur d'entrer un nombre entier naturel puis affiche tous ses diviseurs positifs.

Problème B - Factorielle

Ecrivez un programme qui calcule et affiche la factorielle d'un nombre entier naturel n lut au clavier.
NB : $n! = 1 \times 2 \times 3 \times \ldots \times (n-1) \times n$

Problème C - Somme des nombres lus

Ecrivez un programme qui lit tous les nombres entrés par l'utilisateur et s'arrête puis affiche la somme de ces nombres lorsque l'utilisateur entre le caractère espace.

Coding Time - L'instruction « while »

Problème D - Nombre premier

Écrivez un programme qui lit un entier et indique s'il est un nombre premier ou non.
NB : Un nombre premier est un nombre ayant exactement deux diviseurs positifs : 1 et lui-même.

Problème E - Cent premiers nombre premier

Ecrivez un programme qui affiche les cents premiers nombre premier.

L'instruction « for »

Définition

La boucle for permet d'exécuter un bloc de code un certain nombre de fois comme la boucle while. La différence est qu'elle parcours une séquence. Sa syntaxe est :

```
for variable in sequence :
    bloc d'instructions
```

range(start, stop, step)

La boucle for est très souvent utilisée avec la fonction range. Cette dernière génère une séquence d'entier de "start" (inclus) à "stop" (exclu) suivant le pas "step". Par défaut, start = 0 et step = 1. Ansi range(4) génère la séquence : 0-1-2-3

L'instruction « for »

Exemple 1

```
1 for c in "Bonjour" :
2     print(c)
```

Exemple 2

```
1 for i in range(10, 4, -1) :
2     print(i)
```

Exemple 3

```
1 """ Somme des entiers de 0 à 50 inclus """
2 somme = 0
3 for i in range(51):
4     somme = somme + i
5 print(somme)
```

Coding Time - L'instruction « for »

Problème A - Entre $-n$ et n

Écrivez un programme qui lit au clavier un nombre entier naturel n et qui affiche les entiers relatifs compris entre $-n$ et n.

Problème B - Somme des entiers

Écrivez un programme qui demande un entier n à l'utilisateur et qui lui affiche la somme de tous les entiers naturels strictement inférieur à n.

Problème C - Factorielle

Ecrivez un programme qui calcule et affiche la factorielle d'un nombre entier naturel n lut au clavier.
NB : $n! = 1 \times 2 \times 3 \times \ldots \times (n-1) \times n$

L'instruction pass

Utilité

L'instruction 'pass' est une instruction qui demande à l'interpréteur Python de ne rien faire afin de maintenir une syntaxe correcte.

Exemple

```
if x < 0 :
    pass  # TODO : compléter le code (cas où x < 0)
else :
    print('Traitement du cas où x est positif.')
```

L'instruction break

Utilité

L'instruction 'break' est utilisée pour sortir prématurément d'une boucle 'for' ou 'while'. Cela permet par exemple d'optimiser le code pour rendre l'exécution plus rapide.

Exemple

```
1 for i in range(10) :
2     if i == 4  # condition d'interruption
3         break
4     print(i)
```

L'instruction continue

Utilité

L'instruction 'continue' est couramment utilisée dans les situations où vous souhaitez éviter l'exécution de certaines instructions dans une boucle, mais que vous voulez quand même continuer à itérer.

Exemple

```
1 # ignore tous les nombres pairs
2 for i in range(20) :
3     if x%2 == 0 :
5         continue
6     print(i)
```

Coding Time - Code optimisé

Problème A - Plus grand diviseur mais pas égal

Ecrivez un programme qui affiche le plus grand diviseur d'un nombre entier naturel lu au clavier et qui ne lui est pas égal.

Problème B - Nombre premier

Ecrivez un programme qui lit un entier au clavier et indique s'il est un nombre premier ou non.